(12) UK Patent Application (19) GB (11) 2 338 575 (13) A

(43) Date of A Publication 22.12.1999

(21) Application No 9921267.2

(22) Date of Filing 08.06.1998

Date Lodged 10.09.1999

(30) Priority Data
(31) 09152426 (32) 10.06.1997 (33) JP

(62) Divided from Application No 9812142.9 under Section 15(4) of the Patents Act 1977

(51) INT CL[6]
G06F 9/46

(52) UK CL (Edition Q)
G4A APX

(56) Documents Cited
EP 0362105 A2

(58) Field of Search
UK CL (Edition Q) G4A APX
INT CL[6] G06F
ONLINE:WPI,EPODOC,JAPIO

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Akifumi Nakada
Hajime Tsuchitani
Osamu Furusawa
Toshihiro Suzuki

(74) Agent and/or Address for Service
International Business Machines Corporation
IBM United Kingdom Limited, Intellectual Property Department, Hursley Park, WINCHESTER, Hants, SO21 2JN, United Kingdom

(54) Abstract Title
Message handling

(57) Agents (which may be mobile) in a computer network can hold conversations (similar to human conversations) with each other, in parallel and asynchronously, by sending messages including conversation thread identifiers. If the recipient does not have a conversation thread corresponding to the identifier, it creates one. The messages also provide language type, context and content. A plurality of conversation threads can be halted, a conversation part object capable of controlling them sent to another site through the network, and the conversation threads resumed.

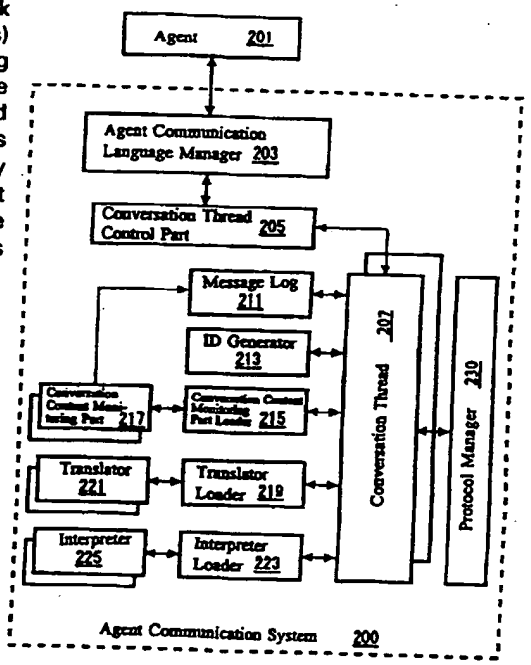


FIG. 3

GB 2 338 575 A

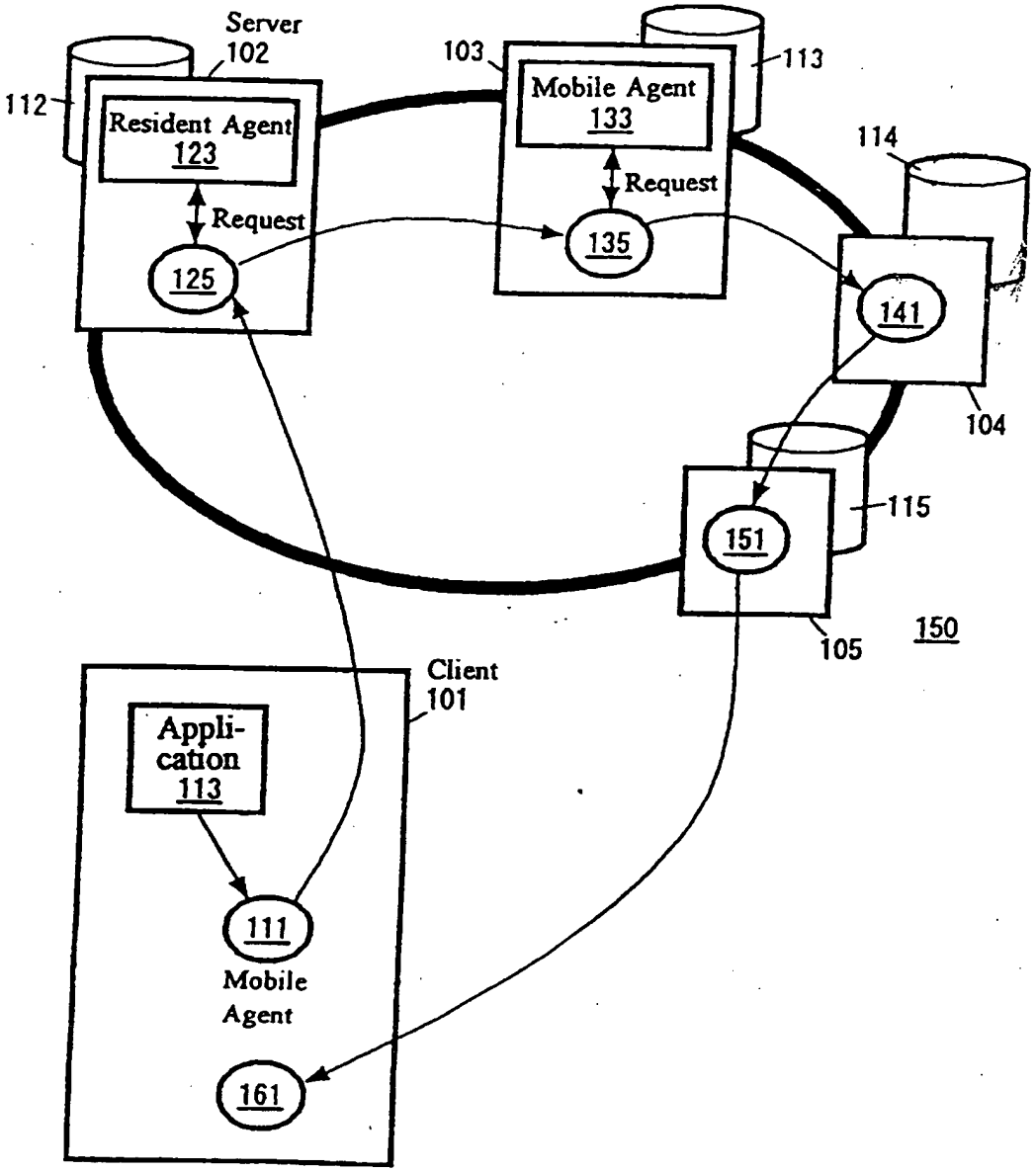

Server
102
112
Resident Agent
123
Request
125
103
Mobile Agent
133
113
Request
135
114
141
104
115
151
105
150
Client
101
Appli-
cation
113
111
Mobile
Agent
161

FIG. 1

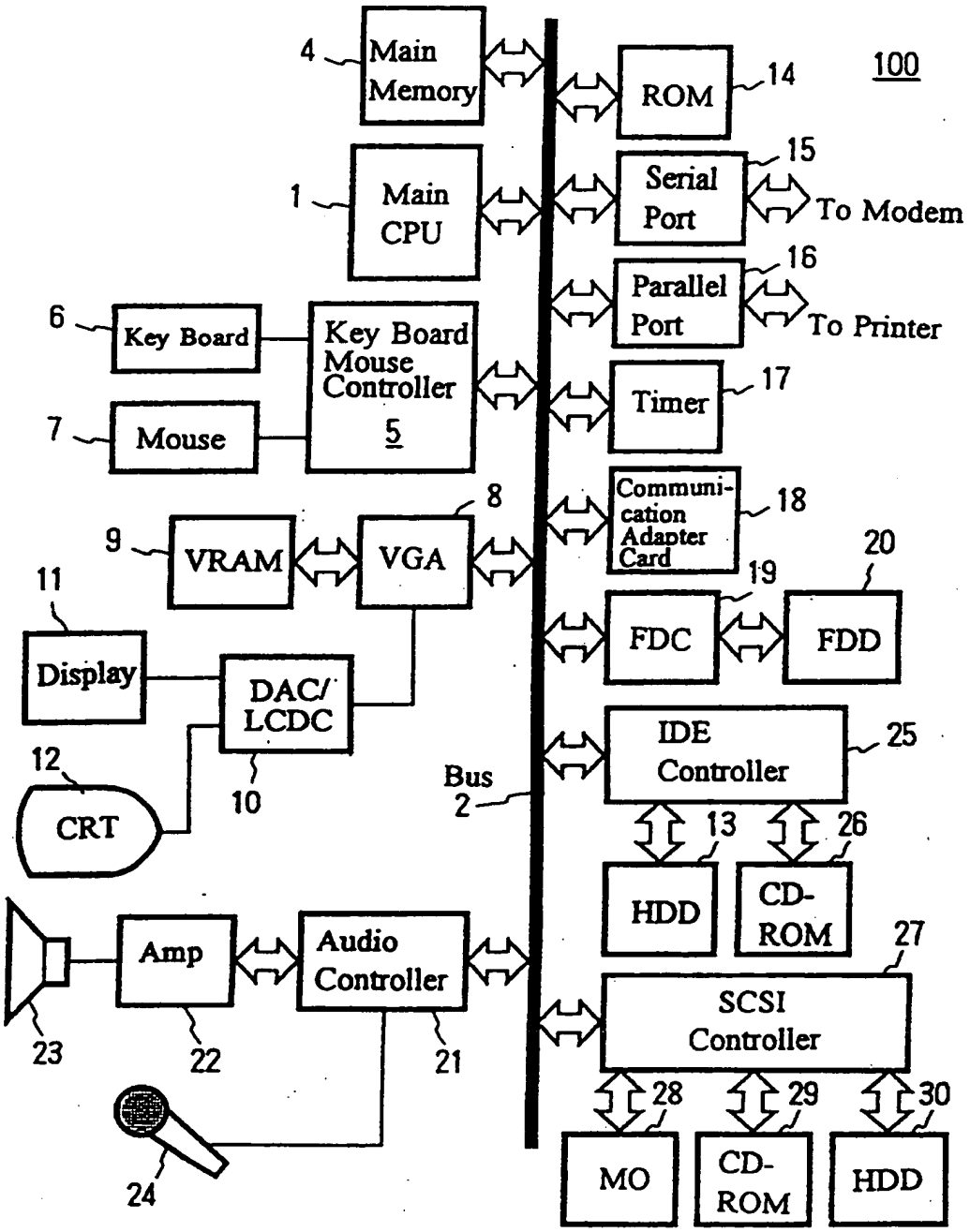

4
Main Memory
ROM
14
100
1
Main CPU
15
Serial Port
To Modem
16
Parallel Port
To Printer
6
Key Board
Key Board Mouse Controller 5
7
Mouse
Timer
17
8
9
VRAM
VGA
Communi-cation Adapter Card
18
20
19
11
FDC
FDD
Display
DAC/ LCDC
IDE Controller
25
12
Bus 2
CRT
10
13
26
HDD
CD-ROM
27
Amp
Audio Controller
SCSI Controller
23
22
21
28
29
30
MO
CD-ROM
HDD
24

FIG. 2

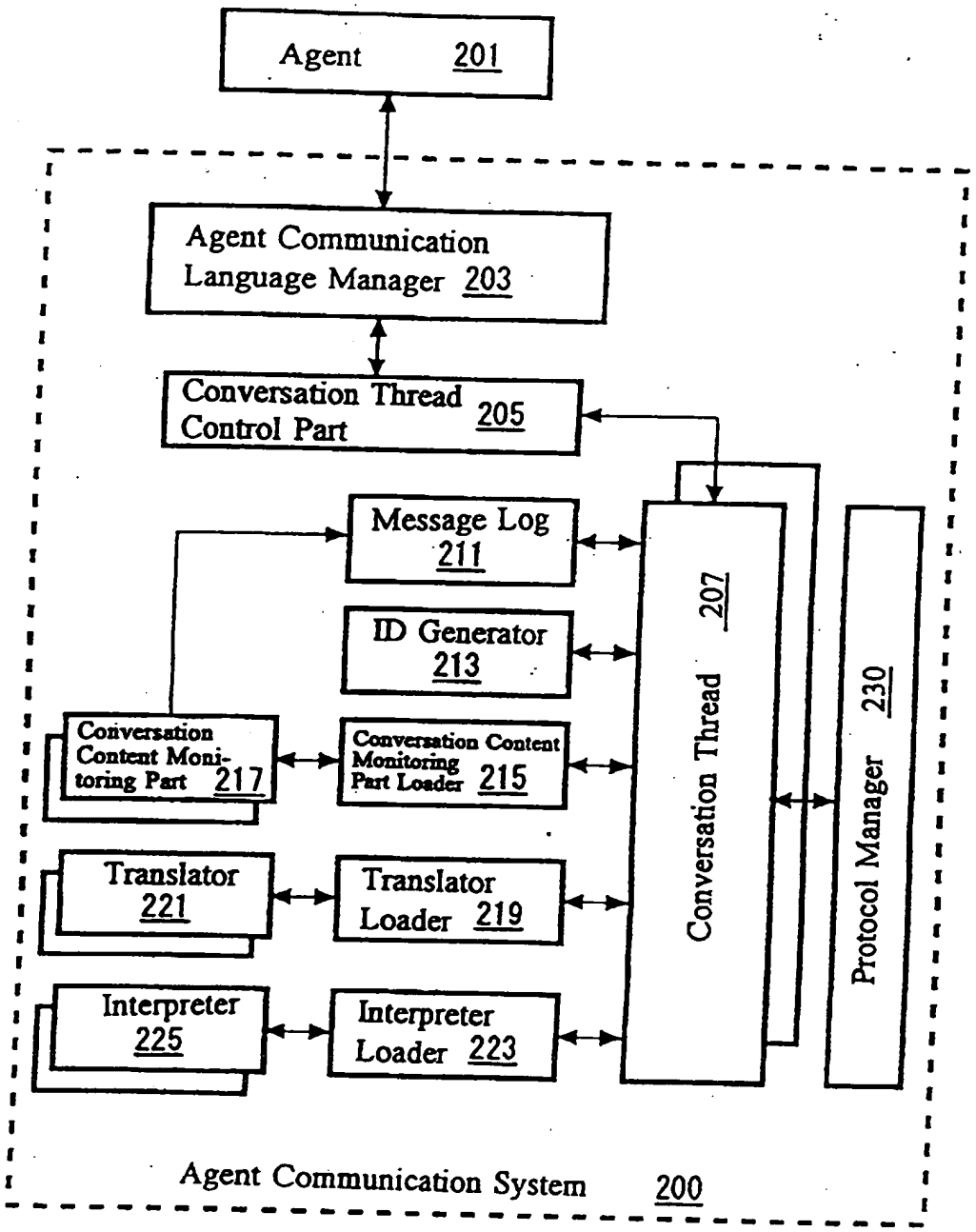

Agent 201
Agent Communication Language Manager 203
Conversation Thread Control Part 205
Message Log 211
ID Generator 213
Conversation Content Monitoring Part 217
Conversation Content Monitoring Part Loader 215
Translator 221
Translator Loader 219
Interpreter 225
Interpreter Loader 223
Conversation Thread 207
Protocol Manager 230
Agent Communication System 200

FIG. 3

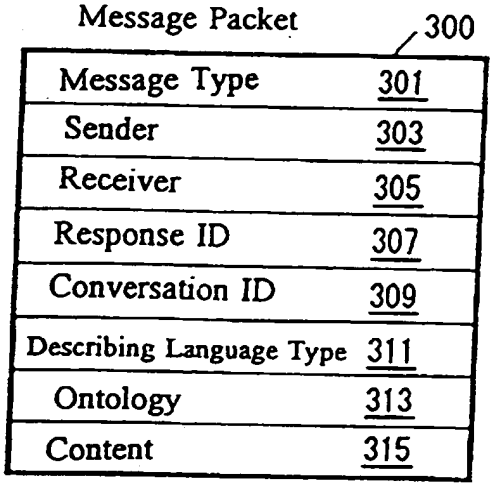

Message Packet
300
Message Type 301
Sender 303
Receiver 305
Response ID 307
Conversation ID 309
Describing Language Type 311
Ontology 313
Content 315

FIG. 4

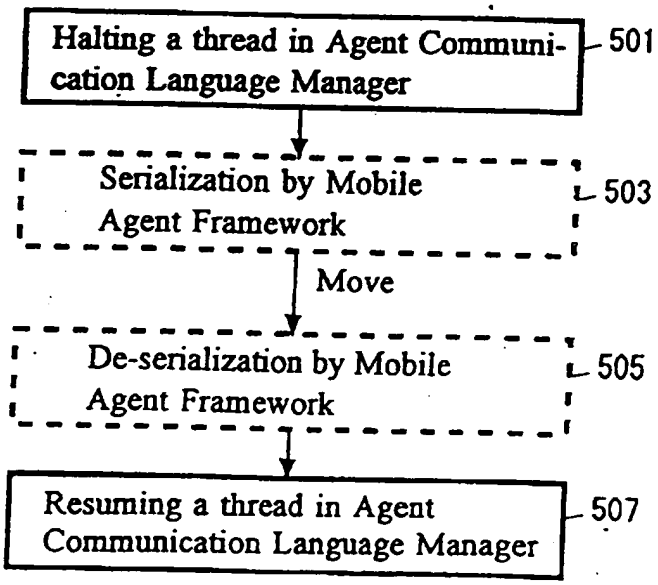

Halting a thread in Agent Communication Language Manager
501
Serialization by Mobile Agent Framework
503
Move
De-serialization by Mobile Agent Framework
505
Resuming a thread in Agent Communication Language Manager
507

FIG. 14

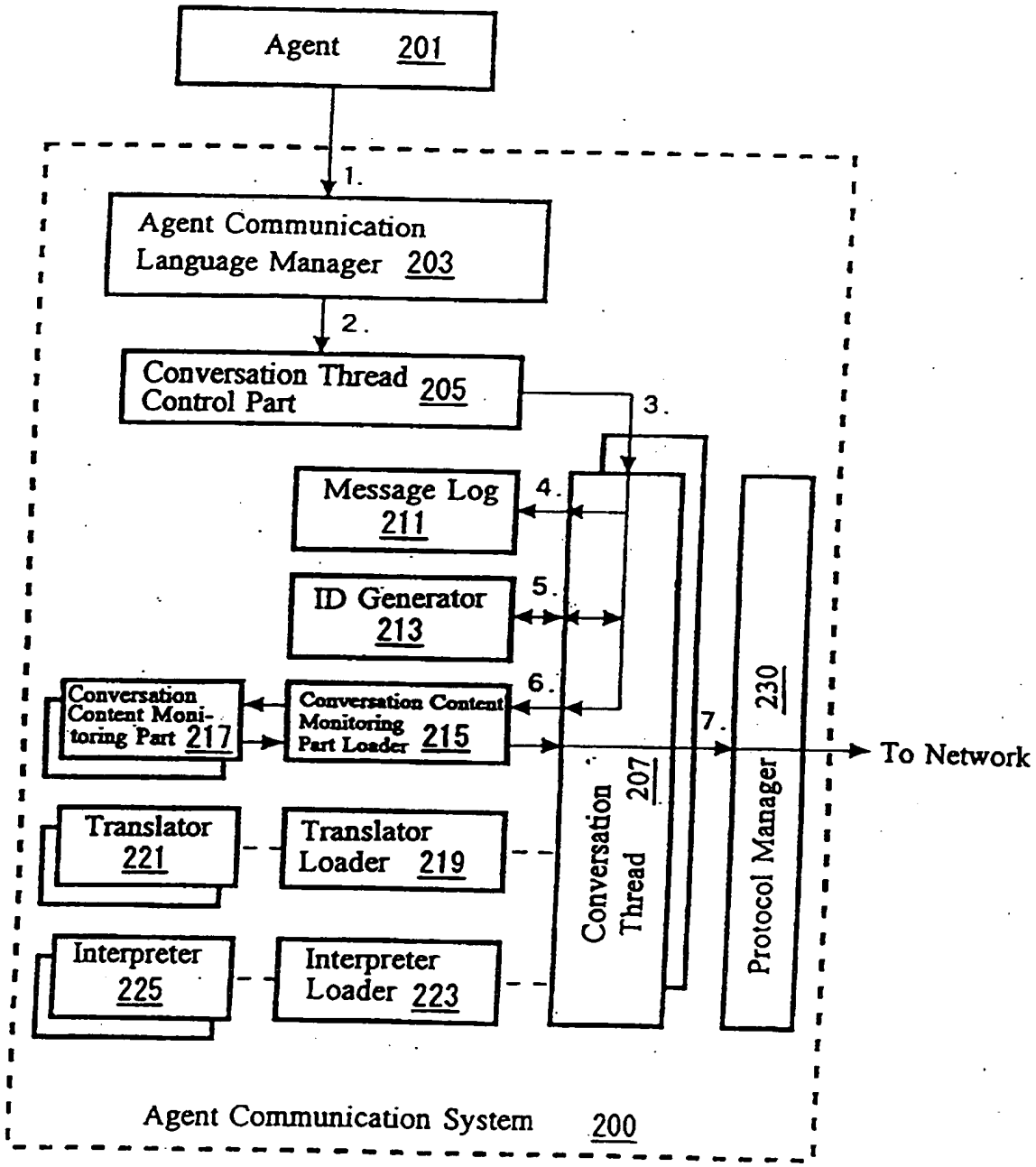
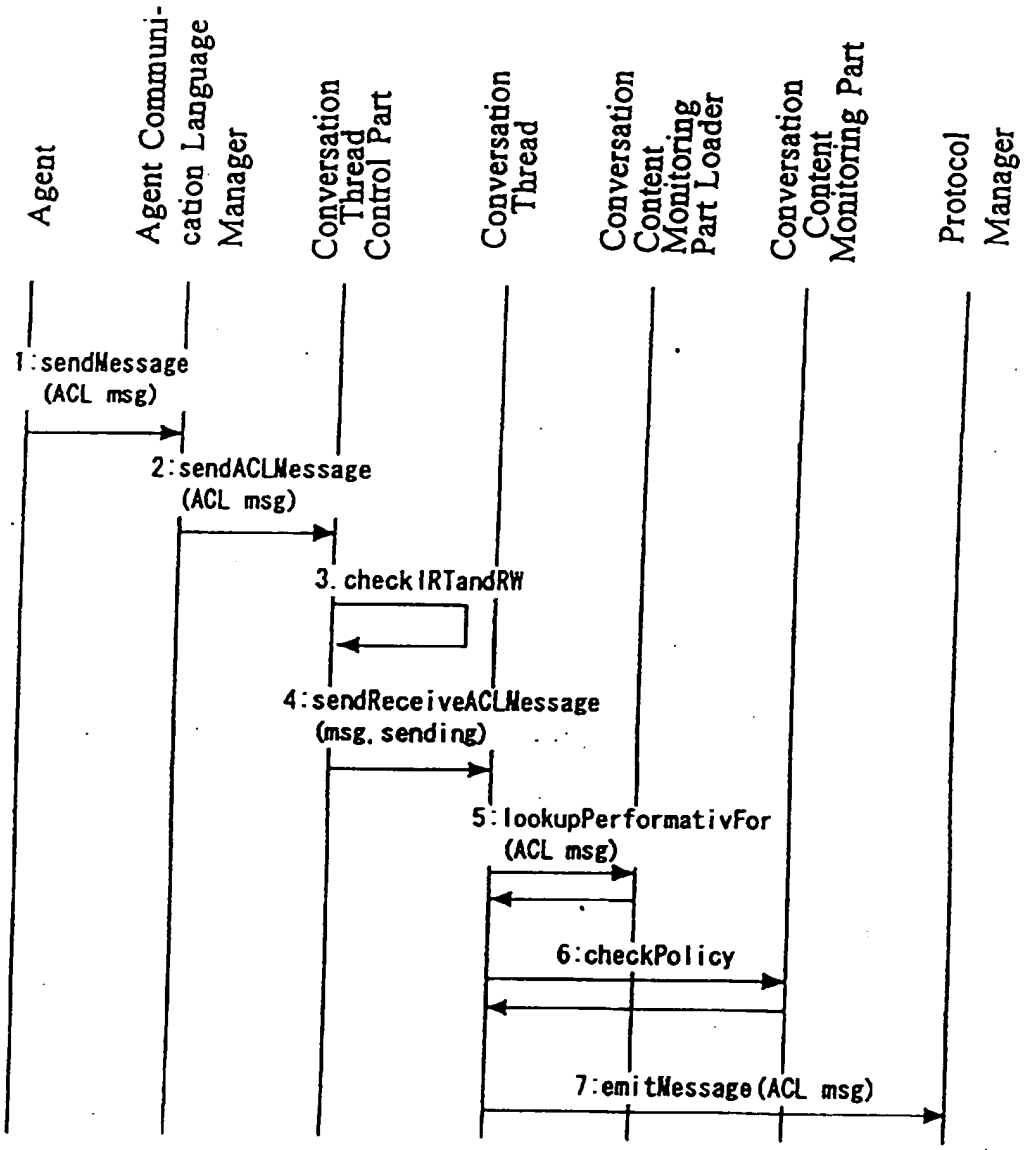

Agent 201
1.
Agent Communication Language Manager 203
2.
Conversation Thread Control Part 205
3.
Message Log 211
4.
ID Generator 213
5.
Conversation Content Monitoring Part 217
Conversation Content Monitoring Part Loader 215
6.
7.
To Network
Translator 221
Translator Loader 219
Interpreter 225
Interpreter Loader 223
Conversation Thread 207
Protocol Manager 230
Agent Communication System 200

FIG. 5



r: Receiver
s: Sender
F: Facilitator
A: Agent
o: Ontology

l: Describing Language Type
c: Contents
rw: Conversation ID
irt: Response ID

FIG. 6

330

Message Type	Conversation Content Monitoring Part
ask-if	ask-if Monitoring Part
sorry	sorry Monitoring Part
⋮	⋮
recruit-all	recruit-all Monitoring Part

FIG. 7

350

Interpreter	Agent Communication Language	Describing Language	Ontology
interpreter A	KQML	Prolog	Stock
interpreter B	FIPA	SL	Stock
⋮	⋮	⋮	⋮
interpreter X	KQML	KIF	Yellow Page

350

FIG. 10

340

Agent Communication Language	Translator
KQML	KQML→FIPA
⋮	⋮
FIPA	FIPA→KIF

FIG. 11

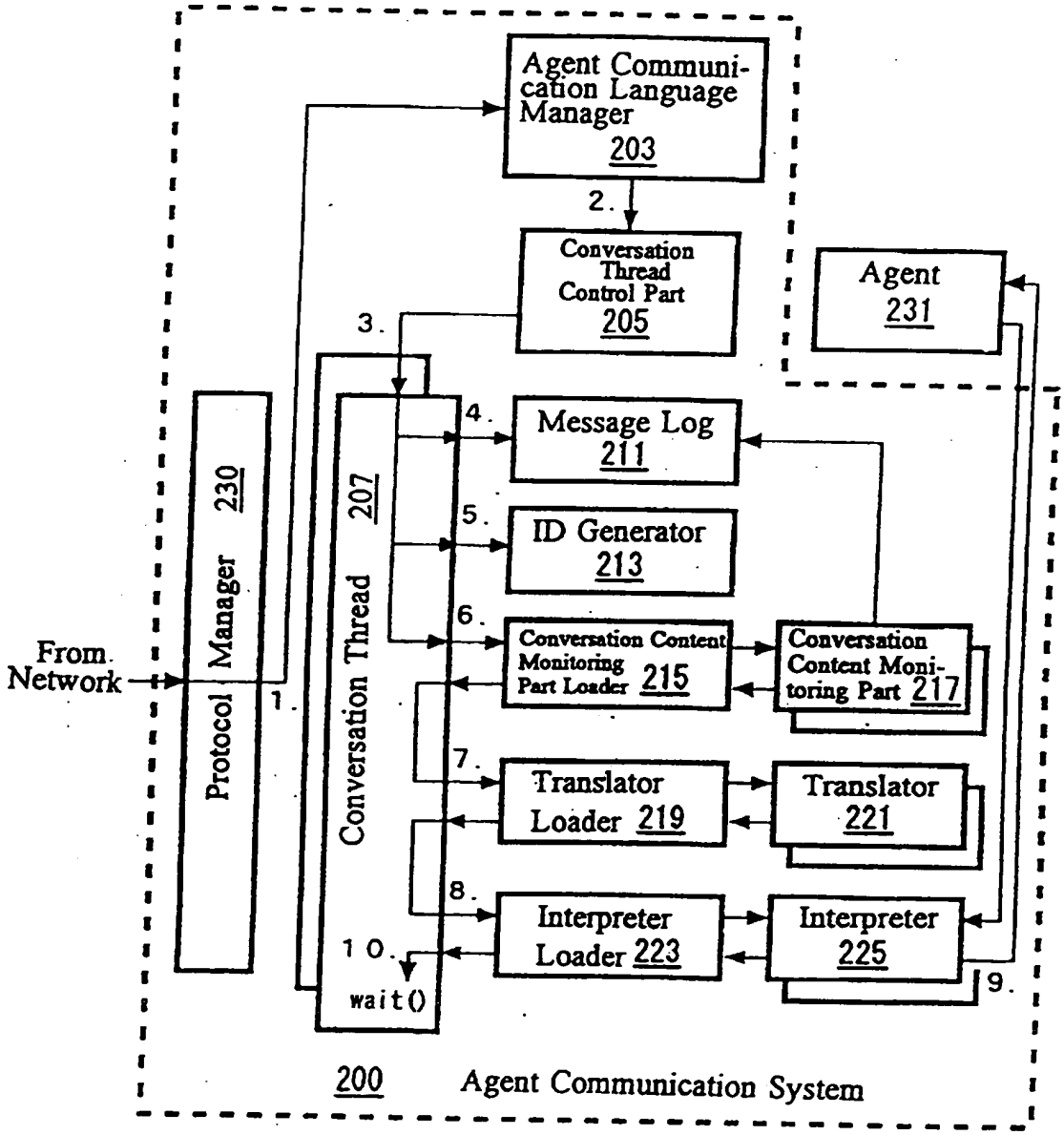
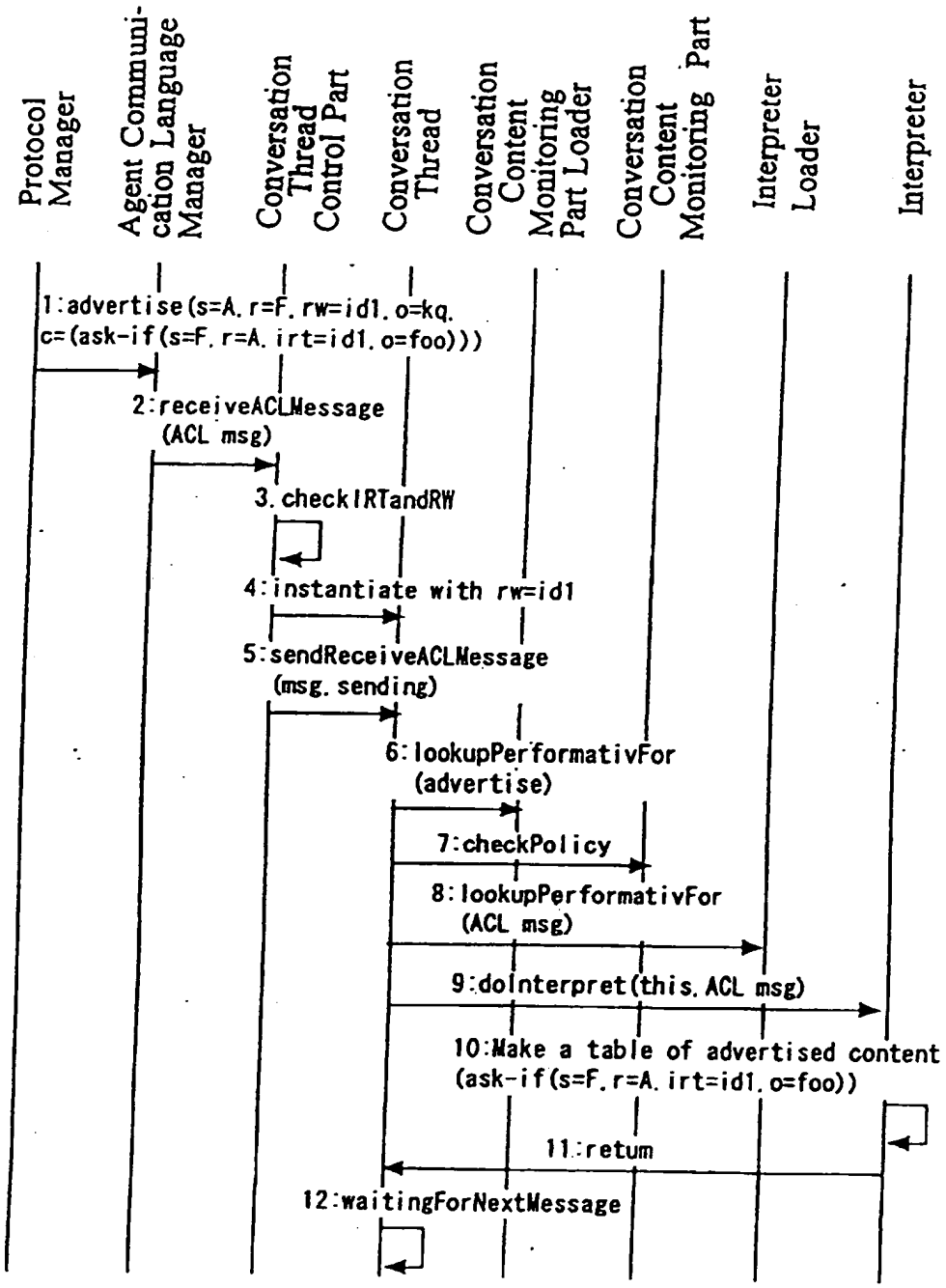

Agent Communication Language Manager 203
2.
Conversation Thread Control Part 205
Agent 231
3.
4.
Message Log 211
5.
ID Generator 213
6.
Conversation Content Monitoring Part Loader 215
Conversation Content Monitoring Part 217
From Network
Protocol Manager 230
1.
Conversation Thread 207
7.
Translator Loader 219
Translator 221
8.
Interpreter Loader 223
Interpreter 225
9.
10.
wait()
200 Agent Communication System

FIG. 8



r: Receiver
s: Sender
F: Facilitator
A: Agent
o: Ontology

l: Describing Language Type
c: Contents
rw: Conversation ID
irt: Response ID

FIG. 9

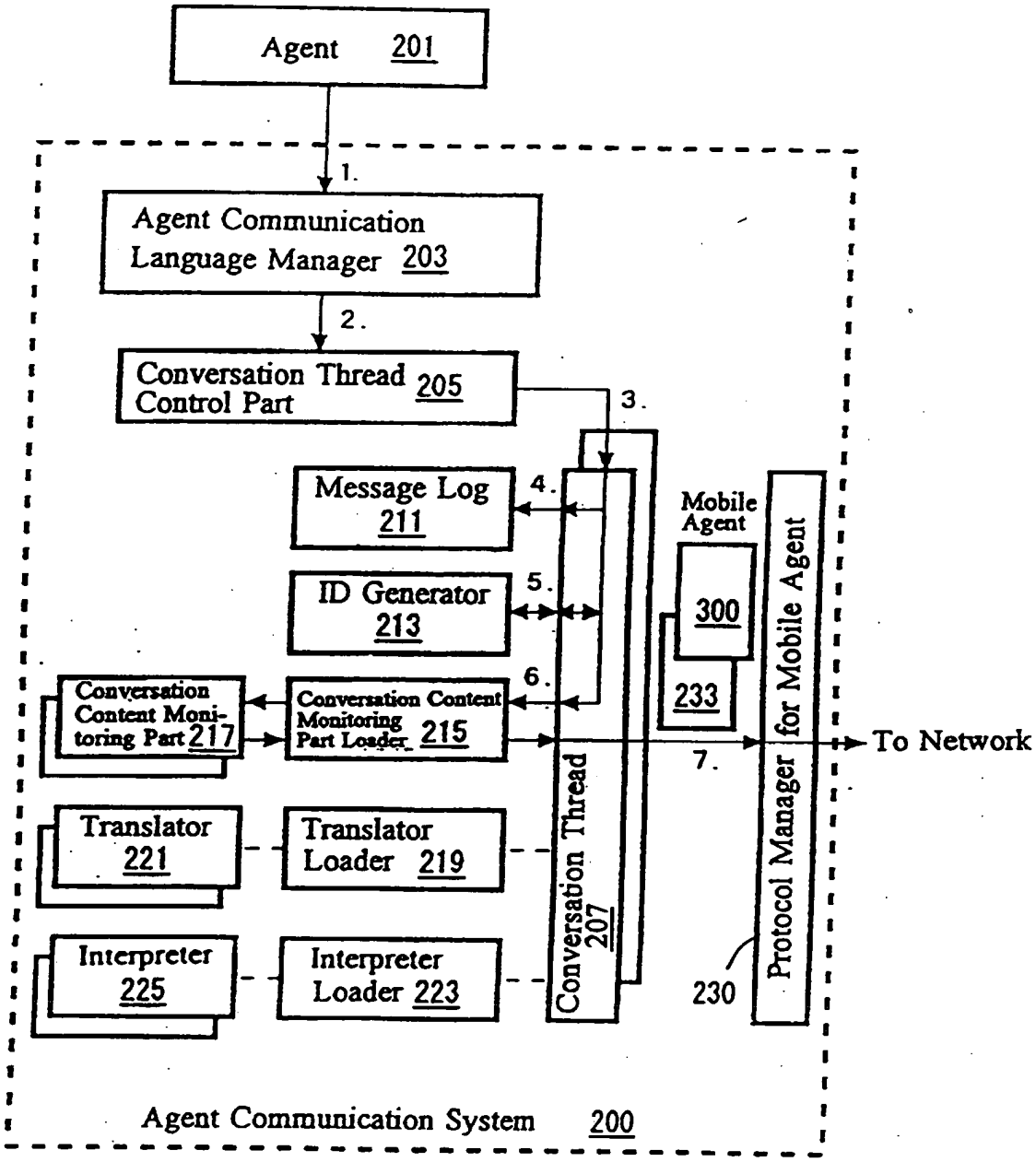

Agent 201
1.
Agent Communication Language Manager 203
2.
Conversation Thread Control Part 205
3.
Message Log 211
4.
ID Generator 213
5.
Conversation Content Monitoring Part Loader 215
6.
Conversation Content Moni-toring Part 217
Translator 221
Translator Loader 219
Interpreter 225
Interpreter Loader 223
Conversation Thread 207
Mobile Agent 300
233
7.
Protocol Manager for Mobile Agent
230
To Network
Agent Communication System 200

FIG. 12

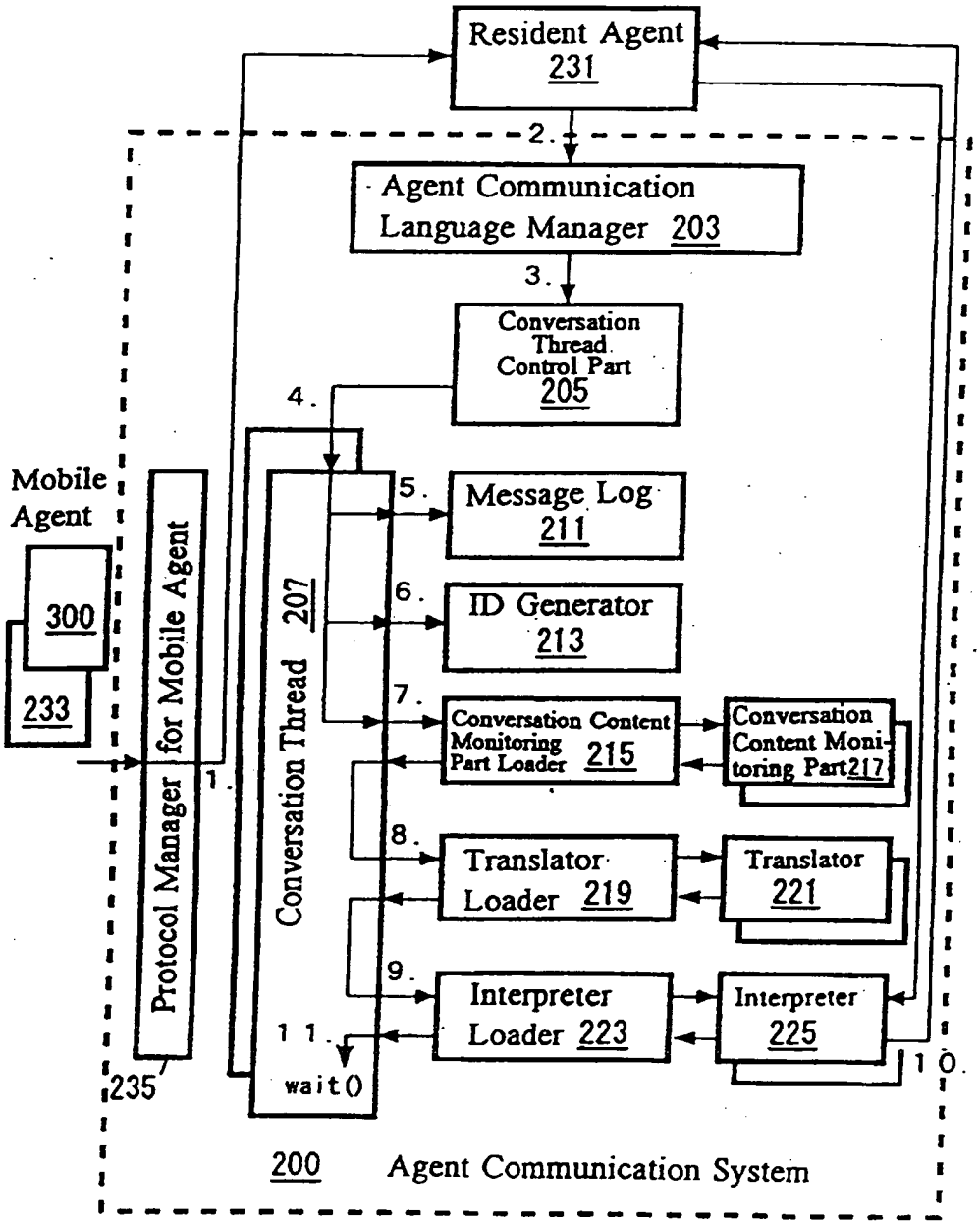
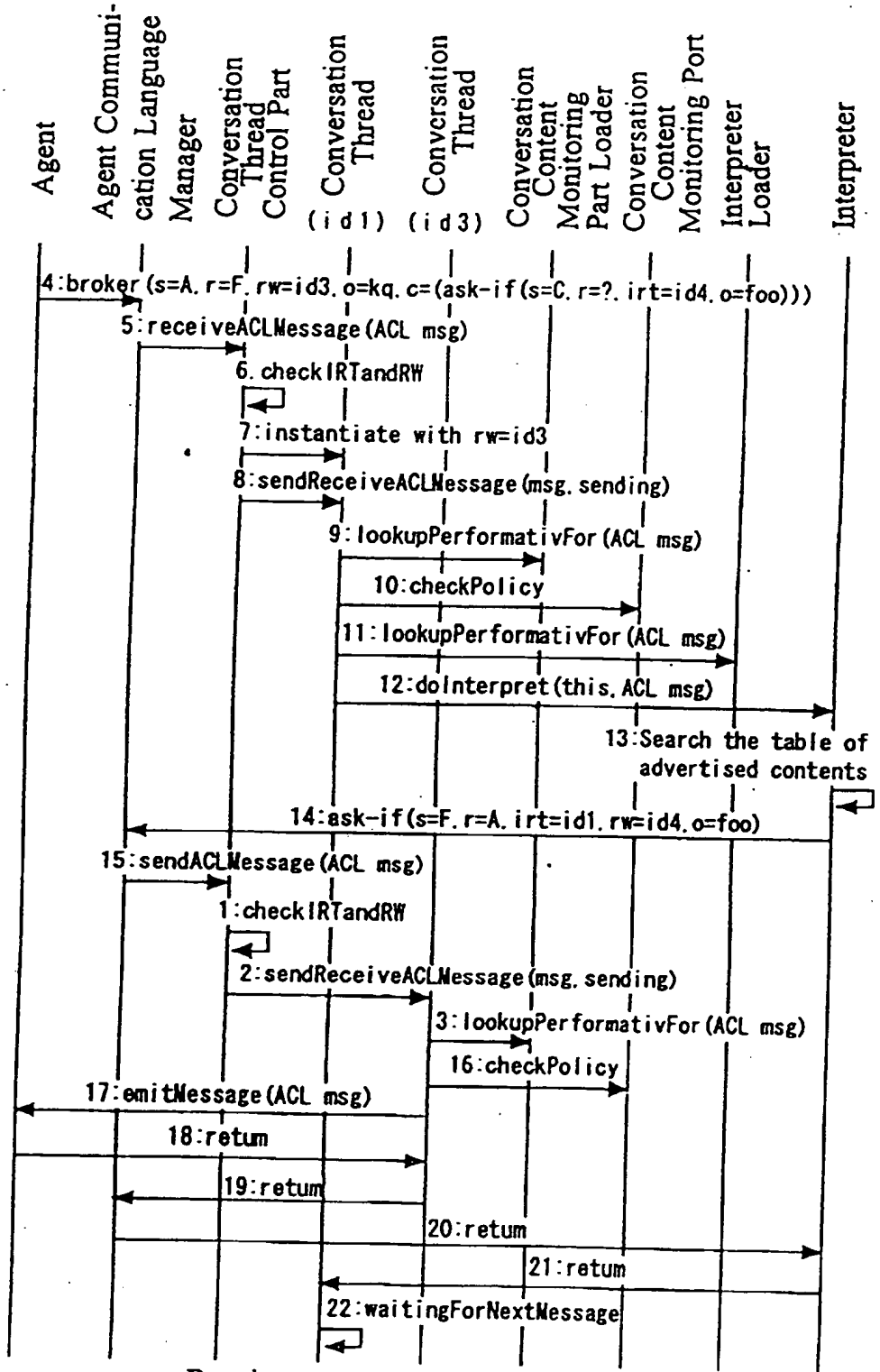

Resident Agent
231
2.
Agent Communication
Language Manager 203
3.
Conversation
Thread
Control Part
205
4.
Mobile
Agent
300
233
Protocol Manager for Mobile Agent
1.
Conversation Thread 207
5.
Message Log
211
6.
ID Generator
213
7.
Conversation Content
Monitoring
Part Loader 215
Conversation
Content Moni-
toring Part217
8.
Translator
Loader 219
Translator
221
9.
Interpreter
Loader 223
Interpreter
225
10.
11.
wait()
235
200
Agent Communication System

FIG. 13



r: Receiver
s: Sender
F: Facilitator
A: Agent
o: Ontology

l: Describing Language Type
c: Contents
rw: Conversation ID
irt: Response ID

FIG. 15

MESSAGE HANDLING METHOD, MESSAGE HANDLING APPARATUS, AND MEMORY MEDIA FOR STORING A MESSAGE HANDLING APPARATUS CONTROLLING PROGRAM

Technological Field of the Invention

This invention relates to a data processing technique and, more particularly, to an improved data processing method for supporting communications among two or more nodes on a network in a distributed computer environment.

Background

There is a mobile agent technology in the art in which an internal condition is moved as it is to a site which is provided in a server existing on a network in a distributed computer environment to form an instruction in the site to which it is moved. Reference is made to USP 5,603,031 (PUPA 7-182174) and Fumihiko Nishida, Susumu Fujiwara et al's "Latest Internet Technology, special edition of Nikkei Communication", pp 104-117, Nikkei BP.

Such mobile agent makes a contact to other agents in the moved-to site (a mobile agent or a resident agent) and may be served as appropriate. A "site" is a location provided by a server existing on the network to which the agent is moved and which supports a contact among agents and absorbs the differences between the hardware and between the platforms.

The mobile agent technology enables a mobile agent to act as a proxy of a human being in handling works such as dynamically adjusting a schedule of internal meetings in compliance with the schedule of attendees and status of reservation of conference rooms and acquiring desired information distributed on the network.

However, in the prior art mobile agents it was difficult to realize a flexible and smooth information exchange and coordination among agents because the communications among agents were realized by a conventional method call (subroutine call) or a low level message passing. Further, it was required to define processing of a stereotype conversation in order to communicate by using an agent, requiring effort on the part of a user of an agent. Furthermore, it was very difficult to communicate between

different types of agents because no agent communication language was used to realize communications among a variety of agents.

A "conversation" between human beings generally consists of steps in which;

1. a speaker prepares a content to be spoken in order to convey the speaker's intent to a listener,

2. the speaker then begins to speak,

3. the listener listens to the speech and understands the flow of the conversation,

4. the listener associates the speech with the background of the speech from what the listener has experienced,

5. the listener understands the content of the speech, and

6. the listener takes an action as a result.

If a messaging system of a computer can be defined which simulates the message processing of a natural language by simulating or approximating this conversation pattern, then it may be possible to process messages which are closer to a natural language than prior art messages, so that an intelligent message system can be constructed which is more user friendly and allows a series of packets having a certain context to be exchanged with the flow of contexts being followed.

There is a desire for a communication system in which an agent (mobile or resident agent) is enabled to communicate with other agents using an agent communication language which is intuitively and easily understandable to a human being.

There is also a desire for a communication system which enables a conversation among agents (mobile or resident agents) to proceed concurrently and asynchronously.

There is also a desire for a communication system in which agents can communicate with each other without being required to be aware of a

variety of different agent communication languages and communication protocols.

There is also a desire to provide a communication system which realizes a high speed parallel conversation processing.

There is also a desire to reduce the work and the time required for developing and maintaining a program as much as possible in providing a support to communication among agents.

It is also desired to provide a message processing system which is capable of flexibly dealing with conversations among agents.

It is also desired to provide a message processing system which reduces the load on a communication network.

Summary of the Invention

According to a first aspect of this invention, a message processing method is provided for execution by a message processor which has a plurality of conversation threads and a conversation part object including conversation thread control part that is capable of controlling said plurality of conversation threads, said message processor being capable of sending a message to another message processor through a network, said method comprising the steps of:

(a) halting said plurality of conversation threads,

(b) sending said conversation part object to another site through said network, and

(c) resuming said plurality of conversation threads at said other message processor.

According to a second aspect of this invention, a media is provided which stores therein a message processing program for execution by a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is capable of sending a message to another message processor through a network, said message processing program comprising:

(a) a program code which instructs said message processor to halt said plurality of conversation threads,

(b) a program code which instructs said message processor to send said conversation part object to another place through said network, and

(c) a program code which instructs said message processor to resume said plurality of conversation threads.

The present invention according to a third aspect provides an agent with an agent communication system (which is preferably implemented in software) which performs functions analogous to the human ear and mouth. In such agent communication system, a "listening" function is provided in the form of API (Application Program Interface) method "handleMessage", while a transmitting function is in the form of API method "sendMessage". In the preferred embodiment of this invention in an object oriented environment, these API's are implemented as a method of an agent communication language manager object of an agent communication system (the method then being written in an object oriented language).

In generating a conversation, an agent prepares an agent communication language (ACL) which is close to the language of a human conversation and sends it by passing it to sendMessage. A message packet contains a sender field and a destination field by which the agent communication system can recognize the location of the conversation partner.

On the other hand, in the node which receives the message packet, the data (message packet contents) from the network is received by a protocol manager which is a component that processes the data coming through the network and which passes it to handleMessage method of the agent communication language manager object. "handleMessage" understands the message packet and executes the content.

According to a fourth aspect of this invention, the agent communication system, upon transmission or receipt of a message packet, determines whether or not a new conversation is initiated and generates a new conversation thread when it is determined that a new conversation is initiated. The conversation thread enables conversations to proceed concurrently and asynchronously among a plurality of agents (mobile or resident agents).

According to another aspect of this invention, functions analogous to an ear and a mouth are provided to an agent simply by instantiating the agent communication language manager object.

According to one aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that is capable of generating a plurality of conversation threads, said method comprising the steps of;

(a) detecting a message packet containing a destination information and a conversation thread identifying information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor,

(c) generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor, and

(d) sending said message packet to the destination which corresponds to said destination information.

According to another aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that is capable of generating a plurality of conversation threads, said method comprising the steps of:

(a) detecting a message packet containing a destination information, a conversation thread identifying information and a content information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor,

(c) generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor,

(d) converting said message packet to a protocol which corresponds to the network to which said message processor is connected, and

(e) sending said message packet to the destination which corresponds to said destination information.

According to still another aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that is capable of generating a plurality of conversation threads, said method comprising the steps of:

(a) detecting a message packet containing a conversation thread identifying information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and

(c) generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to a further aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that is capable of generating a plurality of conversation threads, said method comprising the steps of:

(a) detecting a message packet containing a conversation thread identifying information and a content information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor, and

(c) passing a control to said corresponding conversation thread when it is determined that said corresponding conversation thread exists in said message processor.

According to a still further aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that is capable of generating a plurality of conversation threads, said method comprising the steps of:

(a) detecting a message packet containing a destination information, a conversation thread identifying information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and

(c) generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to a further aspect of this invention, a message processing method is provided for execution by a message processor which is connected to a network and has a conversation thread control part that

is capable of generating a plurality of conversation threads, said method comprising the steps of:

(a) detecting a message packet containing a conversation thread identifying information and a content information,

(b) determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor,

(c) passing a control to said corresponding conversation thread when it is determined that said corresponding conversation thread exists in said message processor, and

(d) analyzing the content of said content information.

According to still further aspect of this invention, a message processor is provided which is connected to a network, said processor comprising:

(a) means for managing a plurality of conversation threads,

(b) an agent communication language manager for detecting a message packet which contains a destination information, a conversation thread identifying information and a content information,

(c) a conversation thread control part (c-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (c-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor, and

(d) a protocol manager for converting said message packet to a protocol which corresponds to a network to which said message processor is connected.

According to a still further aspect of this invention, a message processor is provided which is connected to a network, said processor comprising:

(a) an agent communication language manager for detecting a message packet which contains a conversation thread identifying information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to still a further aspect of this invention, a message processor is provided which is connected to a network and has a conversation thread control part which is capable of generating a plurality of conversation threads, said processor comprising:

(a) an agent communication language manager for detecting a message packet which contains a conversation thread identifying information and a content information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for passing a control to said corresponding conversation when it is determined that said corresponding conversation thread exists in said message processor.

According to still a further aspect of this invention, a message processor is provided which is connected to a network and has a conversation thread control part which is capable of generating a plurality of conversation threads, said processor comprising;

(a) a protocol manager for receiving a message packet containing a conversation thread identifying information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to still a further aspect of this invention, a message processor is provided which is connected to a network and has a conversation thread control part which is capable of generating a plurality of conversation threads, said processor comprising;

(a) a protocol manager for receiving a message packet containing a conversation thread identifying information and a content information, and

(b) an interpreter (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for passing a control to said corresponding conversation thread when it is determined that said corresponding conversation thread exists in said message processor.

According to one aspect of this invention, a recording media is provided which stores thereon a message processing program for execution by a message processor which is connected to a network and has a conversation thread control part capable of generating a plurality of conversation threads, said message processing program comprising:

(a) a program code which instructs said message processor to detect a message packet containing a destination information and a conversation thread identifying information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor,

(c) a program code which instructs said message processor to generate a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor, and

(d) a program code which instructs said message processor to send said message packet to the destination which corresponds to said destination information.

According to a further aspect of this invention, a media is provided which stores thereon a message processing program for execution by a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) a program code which instructs said message processor to detect a message packet containing a destination information, a conversation thread identifying information and a content information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor,

(c) a program code which instructs said message processor to generate a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor,

(d) a program code which instructs said message processor to convert said message packet to a protocol which corresponds to the network to which said message processor is connected, and

(e) a program code which instructs said message processor to send said message packet to the destination which corresponds to said destination information.

According to a further aspect of this invention, a media is provided which stores thereon a message processing program for execution by a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) a program code which instructs said message processor to detect a message packet containing a conversation thread identifying information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor, and

(c) a program code which instructs said message processor to generate a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to a further aspect of this invention, a media is provided which stores therein a message processing program for execution by a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising;

(a) a program code which instructs said message processor to detect a message packet containing a conversation thread identifying information, and a content information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor, and

(c) a program code which instructs said message processor to pass a control to said corresponding conversation thread when it is determined that said corresponding conversation thread exists in said message processor.

According to a further aspect of this invention, a media is provided which stores therein a message processing program for execution by a message processor which has a conversation thread control part

capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising;

(a) a program code which instructs said message processor to detect a message packet containing a conversation thread identifying information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor, and

(c) a program code which instructs said message processor to generate a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to a further aspect of this invention, a media is provided which stores therein a message processing program for execution by a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising;

(a) a program code which instructs said message processor to detect a message packet containing a conversation thread identifying information and a content information,

(b) a program code which instructs said message processor to determine whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exist in said message processor, and

(c) a program code which instructs said message processor to pass a control to said corresponding conversation thread when it is determined that said corresponding conversation thread exists in said message processor, and

(d) a program code which instructs said message processor to analyze the content of said content information.

According to still further aspect of this invention, a media is provided which stores therein a plurality of objects to be loaded to a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) code for managing a plurality of conversation threads,

(b) an agent communication language manager for detecting a message packet which contains a destination information, a conversation thread identifying information and a content information,

(c) a conversation thread control part (c-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (c-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor, and

(d) a protocol manager for converting said message packet to a protocol which corresponds to a network to which said message processor is connected.

According to a still further aspect of this invention, a media is provided which stores therein a plurality of objects to be loaded to a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) an agent communication language manager for detecting a message packet which contains a conversation thread identifying information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

According to still further aspect of this invention, a media is provided which stores therein a plurality of objects to be loaded to a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) an agent communication language manager for detecting a message packet which contains a conversation thread identifying information and a content information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for passing a control to said corresponding conversation when it is determined that said corresponding conversation thread exists in said message processor.

According to still further aspect of this invention, a media is provided which stores therein a plurality of objects to be loaded to a message processor which has a conversation thread control part capable of generating a plurality of conversation threads and is connected to a network, said message processing program comprising:

(a) a protocol manager for receiving a message packet containing a conversation thread identifying information, and

(b) a conversation thread control part (b-1) for determining whether or not a conversation thread corresponding to the conversation thread identifying information contained in said message packet exists in said message processor, and (b-2) for generating a new conversation thread when it is determined that said corresponding conversation thread does not exist in said message processor.

Brief Description of the Drawings

The present invention will now be described in more detail, by way of example, with reference to the accompanying drawings in which:

Fig. 1 show a distributed network environment in which a mobile agent according to an embodiment of this invention operates.

Fig. 2 is a block diagram showing one embodiment of the hardware configuration of a client system or server system implementing this invention.

Fig. 3 is a block diagram of one embodiment of the processing components in the client site or the server site of this invention.

Fig. 4 schematically shows an example of a message packet of this invention.

Fig. 5 is a block diagram of one embodiment of processing components in the client (or server) site of this invention.

Fig. 6 is an object interaction diagram among the components in a preferred embodiment of this invention.

Fig. 7 schematically shows a conversation content monitoring part control table in a preferred embodiment of this invention.

Fig. 8 is a block diagram of one embodiment of processing components in the server (or client) site of this invention.

Fig. 9 is an object interaction diagram among the components in a preferred embodiment of this invention.

Fig. 10 schematically shows an interpreter control table in a preferred embodiment of this invention.

Fig. 11 schematically shows a translator control table in a preferred embodiment of this invention.

Fig. 12 is a block diagram of one embodiment of processing components in the client (or server) site of this invention.

Fig. 13 is a block diagram of one embodiment of processing components in the server (or client) site of this invention.

Fig. 14 is a flow chart showing an embodiment of processing steps taken when the agent communication system of this invention is moved over the network.

Fig. 15 an object interaction diagram among the components in a preferred embodiment of this invention.

Detailed Description of Preferred Embodiments

An embodiment of this invention will now be described with reference to the drawings. With reference to Fig. 1, a distributed network environment 150 is shown which executes mobile objects of this invention. Each server 112 -115 is provided with a site 102-105 for serving the mobile agents 125, 135, 141, etc. The group of sites in the distributed network environment is called a crowd.

An application 113 for forming and moving a mobile agent is in a client system 101. Mobile agents, such as 125 which are sent out by the application 113 may get in contact with other agents(mobile or resident agents) existing in each site 102-105(an agent which gets in contact with a mobile agent to provide a service is called an actor agent), send a request and receive the result of the request. A site provides a support to the contact between agents. Mobile agents, such as 125 may hold the

result of the request received from the actor agent, continue to move, or send the result to the client 101 or other site during movement.

The mobile agent (or a resident agent) 125 may send a message packet to another site and execute a conversation and an instruction in such another site.

Fig. 2 schematically shows a hardware configuration of each node (a server or a client site) existing on the distributed network environment shown in Fig. 1. Each node 100 has a central processing unit(CPU) 1 and a memory 4. The CPU 1 and the memory 4 are connected to a hard disk drive 13 acting as an auxiliary storage via a bus 2. Storage media drives including a floppy disk drive 20, the hard disk drive 13, 30, CD-ROM drive 26, 29, and MO drive 28 are connected to the bus 2 through controllers including a floppy disk controller 19, an IDE controller 25 and an SCSI controller 27.

A portable storage media such as a floppy disk is inserted to the storage media drive such as the floppy disk drive 20. Such floppy disk and other storage media of the hard disk drive 13 and the ROM 14 have recorded therein a computer program which provides instructions to the CPU in cooperation with an operating system to practice this invention, with the program being loaded into the memory 4. The computer software may be compressed or divided into multiple pieces for storage in multiple media.

The node system 100 may be a system which has a user input apparatus, such as a pointing device (a mouse, a joy stick, etc.) 7 or a keyboard 6 for input and a display 12 for presenting a visual data to a user. A parallel port 16 may be provided for connecting a printer. The node system 100 may have a modem connected thereto via a serial port 15 for connection to the network through the serial port 15 and the modem or through a token ring or a communication adapter 18 to communicate with other computer systems.

It will be readily understood that this invention may be implemented in a conventional personal computer(PC), a workstation, a general purpose computer or a combination of them. It should be noted however that these components are chosen as exemplary components and not all components are necessarily an indispensable component of the invention.

For instance, it is not necessary to install a user interface in the server side and it is enough to have a basic data processing function such as a CPU and a memory, and a communication function. It is enough for the client side to have a configuration that is needed to send out a message packet(described later), including a communication function and an input means for designating a message packet and instruct to send out the message packet. Further, when a machine is remote-controlled from a portable terminal and the like, it would be enough to have a data processing function and a communication function to receive an instruction to send the message packet from the portable terminal or the like, detect it and send out the message packet.

The operating system in the client side and the server side may be implemented by one which supports on a standard basis a GUI multi-windows environment such as WindowsNT (trademark of Microsoft), Windows95 (trademark of Microsoft), Windows3.x (trademark of Microsoft), OS/2 (trademark of IBM), X-Window system (trademark of MIT) on AIX (trademark of IBM), and Solaris (trademark of Sun Micro Systems), as well as one which is for character base environment such as PC-DOS (trademark of IBM) and MS-DOS (trademark of Microsoft), and a real time OS such as OS/Open (trademark of IBM) and VxWorks (trademark of Wind River Systems, Inc.) without being limited to any specific operating system environment.

B. System Configuration

With reference to the block diagram in Fig. 3, a system configuration of the server side in the preferred embodiment of this invention is described.

An agent 201 is either a mobile agent which was sent to this server or a resident agent which is resident in this server. The agent 201 may have a dialog with the agent communication system 200 by sending out or receiving a message packet which will be described later.

An agent communication language manager 203 is a component for receiving a message sent from the agent 201 or the existing system and network and for transmitting messages to these systems.

A conversation thread control part 205 is a component for controlling a thread 207 that corresponds to each conversation with an agent like 201, forming a suitable thread in conformance to the flow of

conversation, or giving the control to the thread which has been generated . (multiple conversations are split into multi-threads to accept multiple conversations asynchronously. The thread is called a conversation thread) According to one aspect of this invention, a plurality of such conversation threads are generated by which conversations are held in parallel and asynchronously with a plurality of agents.

A message log 211 is a component for memorizing conversations. A conversation monitoring part 217 is a component which conducts a policy check as to whether or not a conversation is valid from the conversation log stored in the message log and the content of the current message. A conversation content monitoring part loader 215 is a component for calling the conversation monitoring part 217 which corresponds to the message type of the message packet 300.

A translator 221 is a component for translating the agent communication language used in the conversation into another language. A translator loader 219 is a component for controlling the group of translators.

An interpreter 225 is a component for understanding and executing the content of a conversation. A plurality of different such interpreters are provided corresponding to ontologies and describing languages which will be described later, each having a format check routine, a content interpretation routine and an execution routine each specific to the ontology. Take an ontology of traffic reservation for example, assume that a content "R", flight, Narita to Tacoma, 1997/07/06/15:00-1997/07/06/18:00" is received. This is checked for a predetermined format and a specific content interpretation routine is executed, interpreting that "R" is a command which instructs to make a reservation, accessing to a database in a given airline to execute an execution routine for checking the reservation status. Finally, a reservation is made for a non-smoking seat, economy, FlightXX26 for Tacoma, Seattle, departing Narita air port at 17:10 on July 5, 1997 and the reservation is confirmed to the sender.

An interpreter loader 223 is a component for controlling the group of interpreters 225. A protocol manager 230 is a component for converting to a protocol which conforms to the type of network to be connected.

Each of functional blocks in Fig. 3 has been described. These functional blocks (components) which are logical functional blocks are not meant to be implemented as an independent integral hardware or software and may be implemented as a complex of hardware and software or common hardware and software. Further, not all of the functional blocks in Fig. 3 are necessarily a component which is indispensable to this invention.

For instance, it would be enough to install interpretation execution parts such as the conversation content monitoring part 217, the translator 221 and the interpreter 225 because this invention is to provide an agent with a special function to have a conversation with other agents or an existing system. Components such as the conversation content monitoring part loader 215, the translator loader 219 and the interpreter loader 223 are not needed. In one aspect of this invention, because one or more specific number of conversation threads are provided to converse with a specific agent, the agent communication language manager 203, conversation thread control part 205, message log 211 and ID generator 213 are not an indispensable components.

C. Description of Operation

Fig. 4 shows schematically a message packet 300 as used in a preferred embodiment of this invention. Fig. 5 shows an operational condition of each component in Fig. 3 when a message is generated.

In the message type 301 contained in the message packet 300 in Fig. 4, a performative of the agent communication language (ACL) is used in the preferred embodiment of this invention. ACL is a high level language which provides a communication among agents in a form that is close to human language. and includes one provided by FIPA (Foundation for Intelligent Physical Agents), KQML (Knowledge Query Manipulation Language), and KIF (Knowledge Interchange Format). They are laid open by "SEMANTICS FOR AN AGENT COMMUNICATION LANGUAGE, Yannis Labrou, A Doctoral Dissertation for the PhD Defense Examination, Submitted to Defense Committee, at the Computer Science and Electrical Engineering Department (CSEE), University of Maryland Graduate School and "TR CS-97-03, A Proposal for a new KQML Specification, Yannis Labrou and Tim Finin February 3, 1997".

Included in the performative of KQML which is one of ACL are;

ask-if, ask-all, stream-all, eos, tell, untell, deny, insert, uninsert, delete-one, delete-all, undelete, achieve, unachieve, advertise, unadvertise, subscribe, error, sorry, standby, ready, next, rest, discard, register, unregister, forward, broadcast, transport-address, broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all.

A sender 303 is a sender of a message while a receiver 305 is a receiver of a message.

A response ID 307 is an ID which follows a preceding conversation. When an agent has a conversation with a plurality of parties, a plurality of response ID's are allocated corresponding to the conversation threads which will be described later.

A conversation ID 309 is an ID which tells the conversation partner to respond with this ID.

The description language type 311 is information which specifies the language describing the content. The ontology 313 is information which specifies the ontology that the content specifies. In the preferred embodiment of this invention, a ticket purchase ontology, a numerical calculation ontology and a yellow page ontology, etc., are provided.

The content 315 is the particular content of a message and contains a software for forming instructions after it moved to another site in the preferred embodiment of this invention. The content 315 may also internally contain a message packet for movement to another site via a relaying point. Then a From entry and a To entry are provided to indicate a true sender and a final receiver, respectively, in the preferred embodiment of this invention.

An operational condition of each component in Fig. 3 during message generation will now be described with reference to Fig. 5. The agent 201 sends a message packet shown in Fig. 4 to the agent communication system 200. In the preferred embodiment of this invention, the components shown in Fig. 3 are defined as a class of an object oriented language. Each object, upon generating a message, prompts other objects to operate by

sending an instruction to such other objects as shown in the object instruction diagram among components in Fig. 6.

The agent communication language manager 203, upon receiving the message packet 300, determines that the message packet 300 is one for generating a conversation depending on whether it is called with "sendMessage" or "handleMessage" and asks the conversation thread 205 to process.

The conversation thread control part 205 determines whether or not to ask an already generated conversation thread to process with reference to the response ID 307 which is contained in the message packet 300. If the conversation thread 205 determines that it is a new conversation, it generates a new conversation thread 207, instructs the ID generator 213 via the conversation thread 207 to allocate a new conversation ID 309 and moves the control to the conversation thread 207.

In this invention, because the conversation threads are dynamically generated in this way, conversations can be held in parallel and asynchronously with other system(agents or existing system). In the preferred embodiment of this invention, the response ID 307 is allocated with an ID which is uniquely identified on the network, such as "URL + Date + Time + Serial Number".

If a conversation thread 207 corresponding to the response ID 307 has existed, the conversation thread control part 205 determines that this is not a new conversation and moves the control to the corresponding conversation thread 207. If it is desired to continue conversation, the conversation thread 207 instructs the ID generator 213 to allocate conversation ID 309.

The message log 211 memorizes the message packet 300 which is sent to the conversation thread 207.

The conversation content monitoring part loader 216 calls a conversation content monitoring part 217 corresponding to the message type 301 of the message packet 300. The conversation content monitoring part 217 corresponding to the message type 301 may be called by providing a conversation content monitoring part table as shown in Fig. 7 or by using a class name of the conversation content monitoring part which is the same as the message type.

The conversation content monitoring part 217 conducts a policy check as to whether or not a conversation could be valid based on the conversation log stored in the message log 211 and the content of the current message. For instance, if the message type 301 of the received message packet 300 is "sorry" (a message type which means that the content of the conversation is understandable but the receiving party does not have a capability to process), a reference is made to the message log for a message type of the message which has a corresponding response ID and an immediately preceding ID and then it is determined that the message type is of no problem if it is of a type which can be valid as a conversation like "ask-if" a message type which queries about the processing capability of the other party) while it is determined that the message type is in error if it is determined to be of a message type which can not be valid as a conversation like "sorry" and an action is taken such as sending a message packet of "error" a message type which means an error) to the sender 303 of the message packet 300

If it is determined that a conversation can be valid, a message packet 300 is sent from the conversation thread 207 to a node which is designated as a receiver 303 through the protocol manager 230. The protocol manager 230 converts the protocol to one which conforms to the type of the network to be connected so that the agent communication system 200 can send and receive a message packet 300 without being aware of the network protocol.

Now with reference to Fig. 8 and Fig. 9, operational conditions of each component at the node which received the message packet so sent.

The agent communication language manager 203 which has received a message packet 300 from the protocol manager 230 determines that the received message packet 300 is one for receiving message as it is called with "handleMessage" and asks the conversation thread control part 205 to handle it. In the preferred embodiment of this invention, the protocol manager 230 knows the existence of the agent communication language manager 203 as the latter internally instantiates, it can send the incoming data to the agent communication language manager 203.

The conversation thread control part 205 determines whether or not to ask the conversation thread to process the massage packet with reference to the response ID contained in the message packet 300. If the conversation thread control part 205 determines that the packet is a new

conversation, it generates a new conversation thread 207 and transfers it to the conversation thread 207 for processing.

When a conversation thread 207 corresponding to the response ID 307 has existed, the conversation thread control part 205 determines that the packet is not a new conversation and transfers it to the corresponding conversation thread for processing.

The message log 211 memorizes a message packet 300 which is sent to the conversation thread 207.

The conversation content monitoring part loader 215 calls a conversation content monitoring part 217 which conforms to the message type 301 of the message packet 300.

The conversation content monitoring part 217 conducts a policy check as to whether or not a conversation can be valid based on the conversation log stored in the message log 211 and the content of the current message.

If it is determined that a conversation can be valid, the conversation thread 207 transfers the control to the interpreter loader 223 for processing. The interpreter loader 223 understands the agent communication language of the message packet 300 and determines whether or not the expression language type 311 contained in the message packet 300 and an interpreter which corresponds to the ontology 313 are stored in an interpreter control table 350 shown in Fig. 10.

If an interpreter which corresponds to the agent communication language, description language type 311 and the ontology 313 of the received massage packet 300 is stored in the interpreter control table 350, the content 315 is interpreted by that interpreter. If an interpreter which corresponds to the agent communication language, description language type 311 and the ontology 313 of the received massage packet 300 is not stored in the interpreter control table 350, determination is made as to whether or not an interpreter which corresponds to that combination exists in the interpreter library which is managed(accessible) by the system.

If the interpreter exists, it is registered in the interpreter control table 350 and the content 315 is interpreted by using that

interpreter. If an interpreter which corresponds to a combination of the agent communication language, description language type 311 and the ontology 313 does not exist in the interpreter library, determination is made as to whether or not an interpreter which corresponds to a combination of the description language type 311 and the ontology 313 exists in the interpreter library. If it does not exist, it is determined that the message can not be processed. In the preferred embodiment of this invention, the ontologies are hierarchically structured so that an interpreter of a best matching ontology can be used.

If the interpreter exists, the control is transferred to the translator loader 219 to translate the content 315 into the agent communication language of so discovered interpreter. The translator loader 219 makes a reference to the translator control table 340 and determines whether or not there exists a translator which can execute the requested translation. In the preferred embodiment of this invention, if the translator does not exist, the translator library is searched in the way as the interpreter was searched, an entry is made to the table and the translated content 315 is processed by the selected interpreter. If the translator does not exist, it is determined that the message can not be processed.

In the preferred embodiment of this invention, the result of the interpretation by the interpreter 225 is transmitted to the agent 231 where it is processed according to the content of instructions which are described in the content 315. Depending on what is in the content 315, either the conversation thread is destroyed, a process is entered to begin a new message generation, or a new agent is generated. In the preferred embodiment of this invention, the interpreter 225 returns the control to the conversation thread 207. The conversation thread will standby until another message is received.

In contrast to many of API calls in a prior art procedural system, a conversation is asynchronous so that it is unpredictable when the receiving party replies in response to a conversation sent from a sending party to the receiving party. It is possible that information which is possessed by the receiving party could be returned at a timing which is changed by an external force (such as an action by other agents). The reply is sent as a message packet in such case too. The conversation thread 207 waits with wait0 until a reply following the conversation is returned. "wait0" is a method which renders the conversation thread 207

in wait state and waits until it is woken up by a function(method) like "notify0" from others. When the reply is returned, the message is given to the agent system 200. If it is found that the message is a message following the existing conversation, the thread of sleeping conversation is woken up and given the message.

In the preferred embodiment of this invention, when the agent 231 started a conversation in the past and is waiting the reply, that conversation was started through the agent communication system 200(sending "sendMessage" to the agent communication system) and an object "Conversation" is returned as a return value of "sendMessage". The object "Conversation" is associated with an object called "Result" which holds a state. In the preferred embodiment of this invention, the agent 231 can receive a notification by examining the state which is held by the object "Result" through a polling.

In the preferred embodiment of this invention, in order to provide the agent 201 with the agent communication system 200 which is a conversation part of this invention, a coding in JDK (Java Development Kit: Trademark of Sun Microsystems) is used.

The agent communication system 200 of this invention can be provided to the agent 201 simply by instantiating ACLManager by;

```
aclManager = new ACLManager (agentName, protocolName,port);
```

To start a conversation, a kqml object is prepared for example, and the conversation is started simply by ;

```
Conversation Thread conv =
(ConversationThread) aclManager. sendMessage(kqml);
```

When waiting a reply, a polling is conducted by;

```
Result res = conversation. getResult 0;
    if (res. isAvailable0) {do something appropriate}
```

The reply returned or a content of the message sent by other agent is correctly interpreted and executed by an interpreter of the default agent communication system 200.

In the preferred embodiment of this invention so far described, while the interpretation execution parts are registered in the control tables at the time of execution, they need not be registered at the time of execution and, instead, a check may be made as to what interpretation execution parts exist to form and maintain an interpretation execution part control table.

While the message packet 300 moves among nodes in the preferred embodiment of this invention, the message packet 300 may be sent in an attachment to a mobile agent. In such case, the mobile agent 233 moves around sites and holds conversations with the message packet 300 attached thereto. It is further possible to carry out complex jobs such as having a conversation at another site using the process result.

Movement of mobile agent over the network is effected by executing the steps shown in Fig. 14.

The preferred embodiments of the invention provide a number of advantages, as stated below.

As described above, this invention provides a communication system which is capable of communicating between agents (mobile or resident agents) using an agent communication language which is intuitively understandable to a human being.

In one of the aspects of this invention, a message processing system is provided which enables a high speed processing by reducing unnecessary decision logics.

In one of the aspects of this invention, a communication system is provided in which agents can communicate each other without being conscious of a variety of agent communication languages and communication protocols.

In one of the aspects of this invention, a communication system is provided which realizes a high speed parallel conversation processing.

In one of the aspects of this invention, the work and the time required for developing and maintaining a program are reduced as much as possible in providing a support to communication among agents.

In one of the aspects of this invention, a message processing system is provided which is capable of flexibly dealing with conversations among agents.

In one of the aspects of this invention, a message processing system is provided which gives less load to a communication network.

In another aspect of this invention, a message processing system is provided in which resources required for execution are decreased.

CLAIMS

1. A message processing method for execution by a message processor which has a plurality of conversation threads and a conversation part object including a conversation thread control part that is capable of controlling said plurality of conversation threads, said message processor being capable of sending a message to another message processor through a network, the method comprising the steps of:

(a) halting said plurality of conversation threads,

(b) sending said conversation part object to another site through said network, and

(c) resuming said plurality of conversation threads at said another site.

2. A method according to claim 1, wherein said conversation part object is a Java component.

3. A computer program product comprising a computer readable recording media having stored thereon message processing program code for execution by a message processor, the message processing program code including a conversation part object including a conversation thread control part capable of generating a plurality of conversation threads and capable of sending a message to another message processor through a network, said message processing program code further including:

(a) a program code which instructs said message processor to halt said plurality of conversation threads,

(b) a program code which instructs said message processor to send said conversation part object to another network location via said network, and

(c) a program code which instructs said message processor to resume said plurality of conversation threads.

4. A computer program product according to claim 3, wherein said conversation part object is a Java component.

The Patent Office

28

INVESTOR IN PEOPLE

Application No: GB 9921267.2
Claims searched: 1-4

Examiner: Mike Davis
Date of search: 12 October 1999

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): G4A (APX)

Int Cl (Ed.6): G06F

Other: Online: WPI, EPODOC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	EP 0362105 A2 (IBM) eg abstract	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.